A Distributed Protocol for Fractional Stable Paths Problem

Shiva Kintali

College of Computing,
Georgia Institute of Technology,
Atlanta, GA-30332.
Email : *kintali@cc.gatech.edu*

Abstract

The Border Gateway Protocol (BGP) is currently the only interdomain routing protocol deployed in the Internet. BGP can be viewed as a distributed algorithm for solving the *Stable Paths Problem* (SPP) [4]. Not every instance of SPP has a *stable* solution. The most general condition known to guarantee stability of SPP is the *absence of dispute wheel*, proposed by Griffin, Shepherd and Wilfong [4]. They also defined the *Simple Path Vector Protocol* (SPVP), a distributed algorithm for solving SPP. SPVP is sensitive to timing issues and can diverge even when a stable solution exists [4].

Recently, Haxell and Wilfong [5] defined a fractional version of SPP and showed that *every* instance of fractional-SPP (FSPP) has a stable solution. But their proof was non-constructive. In this paper, we define ϵ-*stable solution* of FSPP and present a distributed protocol that *always* converges to an ϵ-stable solution of an FSPP instance, for *any* given $\epsilon > 0$. We define a game-theoretic model for FSPP and present a relation between ϵ-Nash and ϵ-stable solution.

Keywords: border gateway protocol, fractional routing, interdomain routing, path vector protocol, routing games, stable paths problem.

1 Introduction

The Internet consists of smaller networks known as Autonomous Systems. These are subnetworks of routers that are controlled by selfish and competing economic agents. They establish contracts (service level agreements) between each other to transit certain traffic to each other. The task of ensuring connectivity between Autonomous Systems is called *interdomain routing*. The *Border Gateway Protocol* (BGP) is currently the only interdomain routing protocol deployed in the Internet [8].

1.1 Related Work

Griffin, Shepherd and Wilfong [4] introduced *Stable Paths Problem* (SPP) to capture the dynamics of BGP. A *stable* solution to an SPP is an equilibrium point in which each node is assigned its *locally* optimal path to a specific destination node. Routing policies can conflict to cause BGP to diverge, resulting in persistent route oscillations [9]. Hence, not every instance of SPP has a stable solution. Griffin, Shepherd and Wilfong [4] introduced the concept of *dispute wheels* and showed that the absence of dispute wheels implies solvability of an SPP instance. They also presented a distributed algorithm (*Simple Path Vector Protocol* (SPVP)) for solving SPP, and proved that it can never diverge for an SPP instance that has no dispute wheel. A special case of No Dispute Wheel is the *Gao-Rexford* setting ([3, 2]) that depicts the commercial structure underlying the Internet [6]. Recently, Haxell and Wilfong [5] defined a fractional version of SPP and showed that *all* instances of fractional SPP (FSPP) have stable solutions. But their proof was non-constructive. This leads to an interesting open problem as how to design a distributed algorithm to achieve such a fractional stable solution.

1.2 Our Results

In this paper, we define *ϵ-stable solution* of an FSPP instance and present a distributed protocol that *always* converges to an ϵ-stable solution of an FSPP instance, for *any* given $\epsilon > 0$. We define a game-theoretic model for FSPP and present a relation between ϵ-Nash and ϵ-stable solution.

2 SPP and FSPP

2.1 Stable Paths Problem

We denote a network by a simple (no multi-edges and self-loops), undirected and connected graph $G(V, E)$, where V is a set of n source nodes and a unique destination node d, and E is the set of edges. All the source nodes attempt to establish a path to the destination node d. For any node u, $N(u) = \{w \mid (u, w) \in E\}$ is the set of neighbors (a.k.a peers) of u. A *path* from s to t is defined as a sequence of nodes $(v_1, v_2, \ldots v_{k-1}, v_k)$, where $v_1 = s$ and $v_k = t$ and $(v_i, v_{i+1}) \in E$ for $1 \leqslant i \leqslant k-1$. Throughout this paper, we assume that all paths are simple i.e., they do not have repeated nodes. An empty path has no edges and is denoted by ϕ. Let $|P|$ denote the length of P, i.e., number of edges in P. Each non-empty path $P = (v_1, v_2, \ldots v_{k-1}, v_k)$ can be treated as a directed path with the edges directed from its first node v_1 to its last node v_k. We call $P_i = (v_i, v_{i+1}, \ldots, v_k)$ for $1 \leq i \leq k$ a *subpath* of P. Note that P and P_i have the same last node. In particular, $next(P)$ denotes the path $(v_2, \ldots, v_k)$. If $v_2 = v_k$ then $next(P)$ is ϕ. If P and Q are non-empty paths

such that the first node in Q is same as the last node in P, then PQ denotes the path formed by concatenating these paths. We say that path R *ends* with path Q if R can be written as PQ. We call Q as a final segment of R. If Q is non-empty, we call Q as a *proper* final segment of R. Below we present the SPP formalism, defined by Griffin, Shepherd and Wilfong [4].

Permitted Paths : Each source node v would like to establish a connection to d only through one of its trusted paths[1] from v to d. We call these trusted paths as *permitted* paths and denote them by $\mathcal{P}^v$. Let $|\mathcal{P}^v|$ denote the number of permitted paths at node v. We assume that $\forall v \in V$, $\phi \in \mathcal{P}^v$ and we do not count ϕ in $|\mathcal{P}^v|$. If $P = (v, v_1, v_2, \ldots v_k, d)$ is in $\mathcal{P}^v$, then the node v_1 is called the *next-hop* of path P. We assume that $\mathcal{P}^d = \emptyset$.

Ranking Function : For each $v \in V$, there is a non-negative, integer-valued ranking function λ^v (defined over $\mathcal{P}^v$), which represents how node v ranks its permitted paths. If $P_1, P_2 \in \mathcal{P}^v$ and $\lambda^v(P_1) < \lambda^v(P_2)$, then P_2 is said to be *preferred over* P_1. We assume that $\lambda^v(\phi) = 0$ and $\lambda^v(P) > 0$ for each non-empty path $P \in \mathcal{P}^v$.

Strictness of the Ranking Function : If $P_1, P_2 \in \mathcal{P}^v$, $P_1 \neq P_2$ and $\lambda^v(P_1) = \lambda^v(P_2)$, then there is a u such that $P_1 = (v, u)P_1'$ and $P_2 = (v, u)P_2'$. In other words, paths P_1 and P_2 have the same next-hop.

SPP Instance : Let $\mathcal{P} = \{\mathcal{P}^v \mid v \in V - \{d\}\}$. Let $\Lambda = \{\lambda^v \mid v \in V - \{d\}\}$. An instance of the Stable Paths Problem, $\mathcal{I} = \langle G, \mathcal{P}, \Lambda \rangle$, is a graph together with the permitted paths and the ranking functions at each node.

Path Assignment : A path assignment is a function π that maps each node $u \in V$ to a path $\pi(u) \in \mathcal{P}^u$. If a node u is not assigned a path to d then $\pi(u) = \phi$. The set of choices(π, u) of node u is

$$\text{choices}(\pi, u) = \begin{cases} \{(u, v)\pi(v) \mid (u, v) \in E\} \cap \mathcal{P}^u & (u \neq d) \\ \{\phi\} & \text{o.w.} \end{cases}$$

Given a node u, let W be the subset of the permitted paths $\mathcal{P}^u$ such that each path in W has a distinct next hop. Then the *best path in* W is defined to be

$$\text{best}(W, u) = \begin{cases} P \in W \text{ with maximal } \lambda^u(P) & (W \neq \emptyset) \\ \phi & \text{o.w.} \end{cases}$$

Stability : The path assignment π is *stable at node* u if $\pi(u) = \text{best}(\text{choices}(\pi, u), u)$. The path assignment π is *stable* if it is stable at each node u. Any stable path assignment implicitly defines a tree (not necessarily spanning) rooted at d.

Solvability : The Stable Paths Problem $\mathcal{I} = \langle G, \mathcal{P}, \Lambda \rangle$ is said to be *solvable* if there is at least one stable path assignment for $\mathcal{I}$.

[1] These paths are inferred from the service level agreements.

2.2 Fractional SPP

In this section, we present fractional-SPP (FSPP) introduced by Haxell and Wilfong [5]. An instance of FSPP is same as an instance of SPP i.e., $\mathcal{I} = \langle G, \mathcal{P}, \Lambda \rangle$. The only difference between SPP and FSPP is in the definition of their solutions. In a stable solution of SPP, every source node is allowed to have exactly one path to the destination node. In FSPP, nodes are allowed to have multiple paths to the destination. Nodes can assign non-negative weights to their preferred paths, subject to the following conditions. As mentioned in [5], these weights can be interpreted as *fractional routing*.

Feasible Solution : For FSPP, a feasible solution is defined as an assignment of a non-negative weight $w(P)$ to each path $P \in \mathcal{P}^v$, for every v so that the weights satisfy the two properties listed below. For a non-empty path S, let $\mathcal{P}^v_S$ denote the set of paths in $\mathcal{P}^v$ that end with the path S.

- *Unity condition* : For each node v, $\sum_{P \in \mathcal{P}^v} w(P) \leqslant 1$.
- *Tree condition* : For each node v, and each non-empty path S, $\sum_{P \in \mathcal{P}^v_S} w(P) \leqslant w(S)$.

Stable Solution : A stable solution to FSPP is a feasible solution such that for any path $Q \in \mathcal{P}^v$, one of the two following conditions holds:

- $\sum_{P \in \mathcal{P}^v} w(P) = 1$, and each $P \in \mathcal{P}^v$ with $w(P) > 0$ is such that $\lambda^v(P) \geqslant \lambda^v(Q)$.
- there exists a proper final segment S of Q, such that $\sum_{P \in \mathcal{P}^v_S} w(P) = w(S)$, and moreover each $P \in \mathcal{P}^v_S$ with $w(P) > 0$ is such that $\lambda^v(P) \geqslant \lambda^v(Q)$.

3 ϵ-stable Solution

3.1 ϵ-stability

Haxell and Wilfong [5] proved that all instances of FSPP have stable solutions. Their proof works in two stages. In the first stage they show that for any positive constant ϵ, every instance of FSPP has an ϵ-solution. Then they apply a compactness-type argument to conclude that every instance has an exact solution. Below we define *ϵ-stable solution*, which is different from their ϵ-solution. Their definition (of ϵ-solution) is a useful tool in proving that every instance of FSPP has a stable solution. But ϵ-solution is not a *feasible* solution, since it violates tree condition. Our definition of ϵ-stable solution is a feasible solution and is motivated by the widely accepted concept of ϵ-Nash ([1, 10]).

ϵ-stable Solution : An ϵ-stable solution to FSPP is a feasible solution such that for any path $Q \in \mathcal{P}^v$, one of the two following conditions holds:

- $1 - \epsilon \leqslant \sum_{P \in \mathcal{P}^v} w(P) \leqslant 1$, and each $P \in \mathcal{P}^v$ with $w(P) > 0$ is such that $\lambda^v(P) \geqslant \lambda^v(Q)$.

- there exists a proper final segment S of Q, such that $w(S) - \epsilon \leqslant \sum_{P \in \mathcal{P}_S^v} w(P) \leqslant w(S)$, and moreover each $P \in \mathcal{P}_S^v$ with $w(P) > 0$ is such that $\lambda^v(P) \geqslant \lambda^v(Q)$.

Note that, when $\epsilon = 0$, an ϵ-stable solution is equivalent to a stable solution. Below we present a relation between ϵ-Nash and ϵ-stable solution. Our distributed algorithm is a constructive proof showing that all instances of FSPP have an ϵ-stable solution for any given $\epsilon > 0$.

3.2 ϵ-Nash vs ϵ-stable Solution

In this section, we define a game-theoretic model of fractional BGP based on the game-theoretic model for the integral BGP [7]. The main difference between the game-theoretic models for integral and fractional BGP lies in the definition of payoffs to the individual players (source nodes in the network). The ranking function λ^u defines a valuation function that assigns a non-negative integer value to the paths in $\mathcal{P}^u$. It represents the preference over the paths to send traffic to the destination node. Source node u would like to send more (fractional) traffic through a path P with higher $\lambda^u(P)$, subject to the feasibility conditions.

The One-Round Game is a full-information game in which a strategy of a node u is an assignment of weights $w(P)$ for each $P \in \mathcal{P}^u$. Let $w(\mathcal{P}^u) = \{w(P) \mid P \in \mathcal{P}^u\}$. If $w(\mathcal{P}^u)$ is not feasible then u's payoff is zero. If $w(\mathcal{P}^u)$ is feasible, u gets the following payoff :

$$\text{payoff}(u) = \sum_{P \in \mathcal{P}^u} \lambda^u(P) w(P)$$

The observation below follows from the definitions of stable solution and payoff(u).

> Observation : Given the feasible weights $w(\mathcal{P}^v)$ for each $v \in V - \{u, d\}$, payoff(u) is maximized *if and only if* the weights $w(\mathcal{P}^u)$ are stable.

From this observation, it is easy to see that pure Nash equilibria in the One-Round Game correspond to the stable solutions of the FSPP instance. Now we present the relation between ϵ-Nash of the One-Round Game and ϵ-stable solution of FSPP. Given $\epsilon > 0$, ϵ-Nash is a strategy profile such that no player can gain more than ϵ in payoff by deviating from his strategy unilaterally. Similarly, ϵ-stable solution is an assignment of weights to the paths such that no source node can increase (unilaterally) the weight of a preferred path by more than ϵ, without violating the feasibility conditions. In terms of payoffs, a source node u cannot gain more than $\epsilon\lambda^u(P)$ by unilaterally changing $w(P)$. In other words, no source node can gain more than $\epsilon\Gamma$ in payoff by unilaterally changing the weights of (one or more) its preferred paths, where Γ is defined as follows :

$$\Gamma = \max_{v \in V - \{d\}} \{|\mathcal{P}^v| \cdot (\max_{P \in \mathcal{P}^v} \{\lambda^v(P)\})\}$$

Hence, an ϵ-stable solution corresponds to $(\epsilon\Gamma)$-Nash in the One-Round Game.

4 Fractional Path Vector Protocol

As mentioned earlier, not every instance of SPP has a *stable* solution. Also, the Simple Path Vector Protocol (SPVP) is sensitive to timing issues and can diverge even when a stable solution exists [4]. On the other hand, *every* instance of FSPP has a stable solution [5]. Our protocol for FSPP *always* converges to an ϵ-stable solution, for *any* given $\epsilon > 0$. In our protocol, the messages exchanged between neighbors are fractional paths. Hence, we call it Fractional Path Vector Protocol (fractional-PVP).

4.1 The Protocol

As in SPVP, we use a message processing framework which employs a reliable FIFO queue of messages for communication between peers. We model logical time t with discrete values $0, 1, 2, \ldots$. At each discrete time step, one or more nodes with a waiting message runs the $update(u)$ program. This models the asynchronous nature of the Internet. Every node is eventually activated when there is an unprocessed message from one of its neighbors.

We add an artificial node d', connected to the destination node d via a direct link (edge (d, d')). Node d' is always idle and cannot receive or send any messages. We augment all the preferred paths by adding the edge (d, d'). Throughout this section, we assume that all the preferred paths are from the source nodes to d'. Let $\mathcal{Z}$ represent the path consisting of a single edge (d, d'). Note that $\mathcal{Z}$ is a proper final segment of every other permitted path. When a node u changes the weight $w(P)$ of a path $P \in \mathcal{P}^u$, it communicates this change to all its peers. Let $P = (v_1, v_2, \ldots, v_k, d, d')$, where $v_1 = u$, be a path from a node u to the node d'. Let $\mathcal{W}(P) = (w(P_1), w(P_2), \ldots, w(P_k), w(\mathcal{Z}))$, where $P_i = (v_i, v_{i+1}, \ldots, v_k, d, d')$ is a subpath of P. The messages sent by node u to its peers are of the form $\langle P, \mathcal{W}(P)\rangle$, representing the fractional weights of the subpaths of P.

Protocol for the destination node d : In our protocol, the destination node's behavior is different from that of the source nodes. This is in contrast with SPVP, where all nodes run the same program. Destination node d informs its neighbors that it is available *fractionally*. At time t, node d updates $w(\mathcal{Z})$ to $t\epsilon$ and sends the message $\langle \mathcal{Z}, (w(\mathcal{Z}))\rangle$ to its neighbors. This is repeated until $w(\mathcal{Z}) = 1$. This behavior is outlined in the program $increment(d)$. We assume that $\epsilon = 1/\delta$, where δ is an integer. We assume that $w(\mathcal{Z})$ is initialized to 0. Note that, at time $t = \delta + 1$, node d stops sending new messages to its neighbors. We assume that none of the source nodes send messages to d.

---

process *increment*(d)
begin
  if $w(\mathcal{Z}) = 1$ **then**
    exit
  end
  $w(\mathcal{Z}) := w(\mathcal{Z}) + \epsilon$
  foreach $x \in N(d)$ **do**
    send $\langle \mathcal{Z}, (w(\mathcal{Z}))\rangle$ to x
  end
end

```
process update(u)
begin
    let v ∈ N(u)
    let P be a path from v to d
    receive ⟨P, W(P)⟩ from v ⟶
    begin
        let Q := (u, v)P
        if Q ∉ P^u then continue
        if w(P) = w(Q) − ε then
            w(Q) := w(P)
            let R ∈ P^u be the highest ranked ε-unsaturated path such that tightEnd(R) is φ.
            if R ≠ φ then
                w(R) := w(R) + ε
            endif
        else if w(P) = w(Q) + ε then
            do nothing
        else if w(P) = w(Q) + 2ε then
            S := tightEnd(Q)
            if S = φ then
                w(Q) := w(Q) + ε
            else
                let R ∈ P^u_S be the lowest ranked path with positive weight
                if R ≠ Q then
                    w(R) := w(R) − ε
                    w(Q) := w(Q) + ε
                endif
            endif
        endif
        foreach x ∈ N(u) do
            send ⟨Q, W(Q)⟩ and ⟨R, W(R)⟩ to x
        end
    end
end
```

Protocol for the source nodes : Every source node runs the program $update(u)$. As in SPVP, we assume that $update(u)$ is executed in one atomic step and that the communication channels are reliable and preserve message order. The guard (**receive** $\langle P, \mathcal{W}(P)\rangle$ **from** v) is activated when there is an unprocessed message from a neighbor $v \in N(u)$. This causes the message to be deleted from the incoming communication link and processed accordingly. Let $Q = (u, v)P$ be a path ending in P. If $Q \notin \mathcal{P}^u$ then u continues to the next incoming message. If $Q \in \mathcal{P}^u$ then u updates the weights on its preferred paths according to the program. The messages $\langle Q, \mathcal{W}(Q)\rangle$ and $\langle R, \mathcal{W}(R)\rangle$ are sent only if the weights $w(Q)$ and $w(R)$ are modified, respectively. Every node stores the current value of $w(Q)$ for each $Q \in \mathcal{P}^u$.

Let $Q \in \mathcal{P}^u$ be a path from u to d'. Let $Q = (u, v)P$. The *tightend* of path Q is the *longest* path S such that Q ends in S and $\sum_{R\in} \mathcal{P}^u_S\, w(R) = w(S)$. If no such path exists then the tightend

is an empty path, denoted by ϕ. Given the weights of the preferred paths $w(\mathcal{P}^u)$ and $\mathcal{W}(P)$, $tightEnd(Q)$ can be computed locally at the source node u.

In one atomic execution of $update(u)$, the weight of any preferred path is changed by at most ϵ. If $w(P) = w(Q) - \epsilon$ then $w(Q)$ is decreased by ϵ, to preserve the tree-condition. The weight of an ϵ-unsaturated path[2] (with highest rank and empty tightend) is increased by ϵ. If $w(P) = w(Q) + 2\epsilon$ and Q has an empty tightend then $w(Q)$ is increased by ϵ, without changing the weights of the other paths in $\mathcal{P}^u$. If $w(P) = w(Q) + 2\epsilon$ and Q has a non-empty tightend (say S), then the weight of the lowest ranked path with positive weight in $\mathcal{P}^u_S$ is decreased by ϵ and $w(Q)$ is increased by ϵ. More details of the protocol are presented in the proof of Lemma 4.1.

4.2 Proof of Convergence

Let $w(\mathcal{P}^u) = \{w(P) \mid P \in \mathcal{P}^u\}$ and $w(\mathcal{P}) = \{w(\mathcal{P}^u) \mid u \in V - \{d\}\}$. The *network state* is defined as $w(\mathcal{P})$ together with the state of all communication links. A network state is *stable* if all communication links are empty. We present the proof of convergence in two steps. In the first step, we prove that the weights $w(\mathcal{P})$, associated with any stable network state, satisfy the feasibility and the ϵ-stability conditions. In the second second step, we prove that our protocol always converges to a stable network state.

Lemma 4.1. *In fractional-PVP, the weights $w(\mathcal{P})$ associated with any stable network state correspond to an ϵ-stable solution.*

Proof. As in SPVP, we use a message processing framework which employs a reliable FIFO queue of messages for communication between peers. We model logical time t with discrete values $0, 1, 2, \ldots$. At each discrete time step, one or more nodes with a waiting message runs the $update(u)$ program. This models the asynchronous nature of the Internet. Every node is eventually activated when there is an unprocessed message from one of its neighbors. We assume that the program $update(u)$ is executed in one atomic step. For $v \in N(u)$, we define the *pipe* from v to u at time t, $\mathbf{pipe}(u \Leftarrow v, t)$, to be the message queue consisting of messages sent from v to u. Let $Q \in \mathcal{P}^u$ such that $Q = (u, v)next(Q)$. Let $w^t(Q)$ be the weight of path Q at time t. If node u runs $update(u)$ at time t, then $w^t(Q)$ is the weight of Q at the end of its current execution. If node u did not run $update(u)$ at time t, then $w^t(Q) = w^{t-1}(Q)$.

Let t be the current time. Let $\langle next(Q), \mathcal{W}^{t'}(next(Q))\rangle$ be the last message processed from $\mathbf{pipe}(u \Leftarrow v, t')$ for some $t' < t$. Let $\mathcal{W}^t(Q) = (w^t(Q), \mathcal{W}^{t'}(next(Q)))$. Let $\mathcal{W}^t(\mathcal{P}^u) = \{\mathcal{W}^t(Q) \mid Q \in \mathcal{P}^u\}$. We would like to show that $\mathcal{W}^t(\mathcal{P}^u)$ satisfies the feasibility and ϵ-stability conditions for all t. We do this by induction on t. It is easy to see that $\mathcal{W}^0(\mathcal{P}^u)$ satisfies the feasibility and ϵ-stability conditions. Let $\mathcal{W}^{t-1}(\mathcal{P}^u)$ be the weights at time $t-1$. We may assume that they satisfy the feasibility and ϵ-stability conditions. We may assume that there is a waiting message from one of u's neighbor and $update(u)$ is executed at time t.

Let us look at the execution of $update(u)$ at time t. The node u receives the message $\langle P, \mathcal{W}(P)\rangle$ from its neighbor v. Let $Q = (u, v)P$ be a path ending in P. If $Q \notin \mathcal{P}^u$ then u continues to the next incoming message. If $Q \in \mathcal{P}^u$ then u updates the weights on its preferred paths according to the program. The messages $\langle Q, \mathcal{W}(Q)\rangle$ and $\langle R, \mathcal{W}(R)\rangle$ are sent only if the weights $w(Q)$ and $w(R)$ are modified, respectively. In one atomic execution of $update(u)$, the weight of any preferred path is changed by at most ϵ.

[2]Let $P_1 = (u, v)P_2$. P_1 is ϵ-unsaturated if $w(P_2) - w(P_1) > \epsilon$

If $w(P) = w(Q) - \epsilon$ then $w(Q)$ is decreased by ϵ, to preserve the tree-condition. Let $R \in \mathcal{P}^u$ be the highest ranked ϵ-unsaturated path[3] such that $tightEnd(R)$ is ϕ. If $R \neq \phi$ then $w(R)$ is increased by ϵ to preserve ϵ-stability condition. Since $tightEnd(R)$ is ϕ, this does not violate the tree condition. Since $\sum_{P \in \mathcal{P}^u} w(P)$ is not increased, the unity condition is preserved.

If $w(P) = w(Q) + 2\epsilon$ and Q has an empty tightend then $w(Q)$ is increased by ϵ (to maintain the ϵ-stability condition). Since $tightEnd(Q)$ is ϕ, tree condition is not violated. Also, since Q is a path from u to d', the path $\mathcal{Z}$ is not a tightend of Q. Since $w(\mathcal{Z}) \leq 1$, the unity condition is preserved. Since the weights of the other paths in $\mathcal{P}^u$ are not changed, ϵ-stability condition is preserved.

If $w(P) = w(Q) + 2\epsilon$ and Q has a non-empty tightend (say S), then increasing $w(Q)$ would violate the tree condition. It will violate the unity condition if $S = \mathcal{Z}$. To maintain ϵ-stability condition, we need to update the $w(Q)$ by *borrowing* weight from another path. Let $R \in \mathcal{P}^u_S$ be the lowest ranked path with positive weight. If $R \neq Q$, then ϵ-stability condition is violated. Decreasing $w(R)$ by ϵ and increasing $w(Q)$ by ϵ would preserve ϵ-stability. Since $R \in \mathcal{P}^u_S$ the tree condition is not violated. Since $\sum_{P \in \mathcal{P}^u} w(P)$ is not increased, the unity condition is maintained.

Hence $\mathcal{W}^t(\mathcal{P}^u)$ satisfies the feasibility and ϵ-stability conditions at time t, when the source node u processed an incoming message. By induction this is true for all t. Since the communication channels are reliable and preserve message order, in the stable network state (no incoming messages) all the weights on the preferred paths satisfy the feasibility and ϵ-stability conditions at all the source nodes. □

Theorem 4.2. *Fractional-PVP always converges to a stable network state.*

Proof. Let m_1 be a message sent from v to $u \in N(v)$. Let m_2 be a message sent from u to $N(u)$. We say that message m_1 *triggered* m_2 if u received m_1 from v, updated it *weights* (according to the *update*(u) program) and sent m_2 to its neighbors. Note that a single message can trigger multiple messages. For example, in the program *update*(u), the messages $\langle Q, w(Q)\rangle$ and $\langle R, w(R)\rangle$ are triggered by the incoming message $\langle P, w(P)\rangle$. The messages sent by the source nodes are always triggered by an incoming message. The messages sent by the destination node d are not triggered by any messages.

For $v \in N(u)$, we define the *pipe* from v to u at time t, $\mathbf{pipe}(u \Leftarrow v, t)$, to be the message queue consisting of messages sent from v to u. Let $w^t(Q)$ be the weight of path Q at time t. The set of converging nodes, $\mathcal{C} \subseteq V$, are those nodes u such that for some time t and for all $t' \geqslant t$, we have $w^{t'}(Q) = w^t(Q)$ for all $Q \in \mathcal{P}^u$. The oscillating nodes, denoted $\mathcal{O}$, is the set of nodes in V not in $\mathcal{C}$, i.e., $\mathcal{O} = V - \mathcal{C}$. Note that $d, d' \in \mathcal{C}$. Let us assume that fractional-PVP never converges and hence $\mathcal{O} \neq \emptyset$. From the definition of $\mathcal{C}$, there exists a time t_c such that for all $t \geqslant t_c$ and for all $u \in \mathcal{C}$, $w^t(Q) = w^{t_c}(Q)$ for all $Q \in \mathcal{P}^u$. If $u \in \mathcal{C}$ and $x \in N(u)$, then after time t_c no new messages are placed into pipe$(x \Leftarrow u, t)$. Hence there is a time $t_f > t_c$ such that for all times $t > t_f$ all such messages from nodes in $\mathcal{C}$ have been flushed from all communication links.

Let m_2 be triggered by m_1. Node u sends m_2 in response to m_1 only if u detects an ϵ-unsaturated path while executing *update*(u). In other words, if there are no ϵ-unsaturated paths in the network then the protocol converges. Let $u \in \mathcal{O}$, be an oscillating node. We may assume that there exists at least one path $P \in \mathcal{P}^u$, such that $w(P)$ takes at least two different values infinitely often and P is ϵ-unsaturated infinitely often. Let $w(P)$ be assigned k different values infinitely often i.e., $w(P)$

[3]Let $P_1 = (u, v)P_2$. P_1 is ϵ-unsaturated if $w(P_2) - w(P_1) > \epsilon$

is assigned values $\alpha\epsilon$ and $(\alpha + k)\epsilon$ infinitely often (for some $\alpha > 0$). This can happen if one of the following is true :

- There exists a path Q such that $\lambda^u(Q) \geq \lambda^u(P)$ and $w(Q)$ is assigned values $\beta\epsilon$ and $(\beta+k+1)\epsilon$ infinitely often (for some $\beta > 0$).
- There exists a path Q such that P ends in Q and $w(Q)$ is assigned values $\beta\epsilon$ and $(\beta + k + 1)\epsilon$ infinitely often (for some $\beta > 0$).

In both cases there is a path Q such that $w(Q)$ is assigned $k+1$ different values infinitely often. Applying the above argument to $w(Q)$, we find a path R that is assigned $k + 2$ different values infinitely often. We can continue this argument for R and so on. Since δ and $|V|$ are finite and all the preferred paths are simple, this leads to a contradiction. This implies that none of the weights take two different values infinitely often. Hence, fractional-PVP converges to a stable network state in finite time. □

Corollary 4.3. *Fractional-PVP always converges to an ϵ-stable solution of an FSPP instance, for* any *given $\epsilon > 0$.*

5 Conclusion

We defined *ϵ-stable solution* of an FSPP instance and presented a distributed protocol that *always* converges to an ϵ-stable solution of an FSPP instance, for *any* given $\epsilon > 0$. We defined a game-theoretic model for FSPP and presented a relation between ϵ-Nash and ϵ-stable solution.

Acknowledgements : I am grateful to H. Venkateswaran for many helpful and motivating discussions throughout the course of this project. I would like to thank Milena Mihail for initial discussions on BGP. This research is partially funded by ARC Thinktank Fellowship at College of Computing of Georgia Institute of Technology. I would like to thank Gordon Wilfong and Penny Haxell for sending a preprint of their paper [5]. I would like to thank Gordon Wilfong, Michael Schapira and Vijay Ramachandran for helpful discussions during a recent DIMACS workshop[4], where this research was presented as work-in-progress. I would like to thank Gordon Wilfong for helpful comments on initial drafts of this paper.

References

[1] R.J. Aumann. Acceptable points in general cooperative n-person games. *In: A.W. Tucker, R.D.Luce (Eds.), Contributions to the Theory of Games, Annals of Mathematical Studies 40, Princeton University Press, Princeton, New Jersey, 1959.*

[2] Lixin Gao, Timothy G. Griffin, and Jennifer Rexford. Inherently Safe Backup Routing with BGP. *In 20th IEEE INFOCOM*, pages 547–556, 2001.

[3] Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. *IEEE/ACM Transactions on Networking*, 9(6):681–692, 2001.

[4]DIMACS/DyDAn Workshop on Secure Internet Routing, March 24 - 26, 2008, Rutgers University

[4] T. Griffin, F. B. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking*, 10(2):232–243, 2002.

[5] P. E. Haxell and G. T. Wilfong. A fractional model of the border gateway protocol. *SODA*, 2008.

[6] Geoff Huston. Interconnection, Peering and Settlements. *In Internet Global Summit (INET). The Internet Society*, 1999.

[7] Hagay Levin, Michael Schapira, and Aviv Zohar. Interdomain routing and games. *STOC*, 2008.

[8] Y. Rekhter and T. Li. A border gateway protocol. RFC 1771 (bgp version 4). 1995.

[9] K. Varadhan, R. Govindan, and D. Estrin. Persistent route oscillations in inter-domain routing. *ISI technical report 96-631, USC/Information Sciences Institute*, 1996.

[10] H. Peyton Young. Cost allocation, demand revelation, and core implementation. *Mathematical Social Sciences*, 36:213228, 1998.